**First,**

void SEFInfoImpl::load() {

...

file->position(file->size() - 8);

auto sefTailStartSignatureOffset = file->read<int>() + 8;

// adjust for HEIC 'sefd' box header

if (file->mimeType() == IMAGE\_HEIC) {

sefTailStartSignatureOffset -= 8;

}

file->position(file->size() - sefTailStartSignatureOffset);

...

}

Or,

auto tailOffset = file->read<int>();

if (file->mimeType() == IMAGE\_HEIC) {

tailOffset -= 8; // adjust box header

}

auto sefTailStartSignatureOffset = tailOffset + 8;

**Second,**

// HEIFMetaReader.cpp (add/replace the two methods)

Metadata HEIFMetaReader::getXMP() {

try {

const int coverId = getCoverImageId();

// HEIF 'mime' item is used for XMP in your writer; mirror it here

auto meta = getMetadataInfo<MIME>(coverId); // SHEIFD\_METADATA\_INFO { data, data\_size, ... }

if (meta.data == nullptr || meta.data\_size <= 0) return {};

// cache bytes so Metadata::data() can read them later

auto cache = std::make\_shared<std::vector<uint8\_t>>(

static\_cast<uint8\_t\*>(meta.data),

static\_cast<uint8\_t\*>(meta.data) + meta.data\_size

);

Metadata m{};

m.type = META\_XMP;

m.position = 0;

m.payload = cache->size();

m.offset = 0;

m.byteReader.read = [cache](size\_t pos, size\_t len) -> byte\_arr\_type {

// clamp safely

const size\_t n = cache->size();

if (pos > n) return {};

const size\_t take = std::min(len, n - pos);

return byte\_arr\_type(cache->begin() + pos, cache->begin() + pos + take);

};

return m;

} catch (const std::exception& e) {

ALOGE("getXMP failed: %s", e.what());

return {};

}

}

Metadata HEIFMetaReader::getExif() {

try {

const int coverId = getCoverImageId();

auto meta = getMetadataInfo<EXIF>(coverId); // raw Exif item bytes

if (meta.data == nullptr || meta.data\_size <= 0) return {};

auto cache = std::make\_shared<std::vector<uint8\_t>>(

static\_cast<uint8\_t\*>(meta.data),

static\_cast<uint8\_t\*>(meta.data) + meta.data\_size

);

// If your downstream EXIF reader expects the TIFF payload only (not the "Exif\0\0" wrapper),

// compute an offset here. Otherwise keep offset=0 to return bytes as-is.

size\_t exifOffset = 0;

if (cache->size() >= 6) {

// Many HEIF encoders store Exif as: "Exif\0\0" + TIFF

if (std::equal(cache->begin(), cache->begin() + 4, "Exif") && (\*cache)[4] == 0 && (\*cache)[5] == 0) {

exifOffset = 6;

}

}

Metadata m{};

m.type = META\_EXIF;

m.position = 0;

m.payload = cache->size();

m.offset = exifOffset;

m.byteReader.read = [cache](size\_t pos, size\_t len) -> byte\_arr\_type {

const size\_t n = cache->size();

if (pos > n) return {};

const size\_t take = std::min(len, n - pos);

return byte\_arr\_type(cache->begin() + pos, cache->begin() + pos + take);

};

return m;

} catch (const std::exception& e) {

ALOGE("getExif failed: %s", e.what());

return {};

}

}